KU LEUVEN

FACULTY OF ECONOMICS AND BUSINESS

Resource loading with time windows

Talla Nobibon F, Leus R, Nip K, Wang Z.

KBI_1402

Resource loading with time windows

Fabrice Talla Nobibon[*], Roel Leus[†], Kameng Nip[‡], Zhenbo Wang[‡]

January 20, 2014

Abstract

Resource loading appears in many variants in tactical (mid-term) capacity planning in multi-project environments. It deals with the development of a rough sketch of the resource usage and timing of the work packages of a portfolio of orders. The orders need to be executed within a time horizon organized into periods, where each period has a known number of workers available. Each order has a time window during which it must be executed and an upper and a lower bound on the number of workers that can work on that order in a period. The duration of the order is not fixed beforehand but depends on the number of workers (intensity) with which it is executed.

In this article we define three fundamental variants of resource loading and we study six special cases that are common to the three variants of the problem. We present algorithms for the cases that can be solved either in polynomial time or in pseudo-polynomial time. We prove that the remaining cases are NP-complete in the strong sense and we also discuss the existence of approximation algorithms for some of these cases. Finally, we comment on the validity of our results when orders must be executed without preemption.

Keywords: Resource loading; dynamic programming; NP-complete; preemption; approximation algorithms.

1. Introduction

Many organizations exploit a project structure to cope with large and highly complex tasks. These tasks usually involve expertise from many different departments or groups, such as engineering, various production departments, process and production planning, and service (De Boer, 1998). Traditionally, research in the area of project planning has focused on planning isolated projects. Many companies, however, adopt an organizational structure in which multiple projects are run in parallel and share the same scarce resources. This corresponds with a so-called 'matrix structure': resources are associated with functional departments but are assigned to different ongoing projects throughout time (Larson and Gobeli, 1989). Adler et al. (1995) point out that frequent conflicts of interest arise when more than one project requires the same resource at the same time. The global coordination of such multi-project organizations is the essence of multi-project management. An aggregate, combined project plan facilitates cross-project analysis and reporting (Kerzner, 1998), and it is a good help for management to ensure that the organization does not take on more projects than it can complete (cfr. Wheelright and Clark, 1992). Speranza and Vercellis (1993) observe that, because of the multi-objective and dynamic nature of multi-project planning, a monolithic approach

[*]Senior Operations Research Analyst, FedEx Europe, Middle East, Indian Subcontinent & Africa

[†]ORSTAT, Faculty of Economics and Business, KU Leuven, Belgium. E-mail: Roel.Leus@kuleuven.be

[‡]Department of Mathematical Sciences, Tsinghua University, Beijing, China

is inadequate for most practical applications, and they suggest a decomposition into an operational and a tactical planning phase. In the latter phase, which we refer to as *resource loading*, the distinct production operations are often aggregated into jobs representing whole production phases, and planning and scheduling typically work on these so-called *work packages* (Alfieri et al., 2011). While detailed production planning (scheduling) at the operational level has already been the subject of vast research efforts for various industrial sectors, the work on mid-term capacity planning is rather limited. This paper contributes to the better understanding of the latter problem through a careful analysis of the underlying combinatorial optimization problem.

Problem description. We are given a set of candidate orders (henceforth, 'jobs') and a workforce roster for the planning horizon. The workforce capacity may change over time as a function of the individual workers' calendars and due to earlier capacity-allocation decisions. Each job has a *release time* and a *deadline*, which together make up the job's *time window*, and also a *work content* (equivalently referred to as *workload*), expressed in man-periods. The essential difference between 'classic' planning and scheduling, on the one hand, and resource loading, on the other hand, is that in the latter case the resource utilization of the jobs to be performed is still flexible, i.e., the workload can, up to a certain extent, be spread over several periods of the planning horizon, and this in varying proportions. The duration of a job is therefore not fixed beforehand but depends on the number of workers (intensity) with which the job is executed. Additionally, these intensities can change over time. A job can, for example, be started with two workers and completed with one worker, or vice versa. This model is of particular interest to tactical planning in equipment maintenance and in project-based make-to-order (MTO) and engineer-to-order (ETO) environments, where capacity utilization can strongly fluctuate over time and orders may vary significantly with respect to routings, material and tool requirements, and these attributes may moreover not be fully known at the stage of order acceptance. In the classic product-process matrix of Hayes and Wheelright (1979), these environments belong to the same cell because they all deal with low-volume, low-standardization, one-of-a-kind products or assignments, with high variety. The available capacity is therefore flexibly planned in an aggregate fashion, relegating detailed scheduling decisions to the operational level. Throughout the text, we mainly refer to workers as the resource type being planned, but the results obtained obviously also directly apply to other renewable (i.e., non-consumable) resource types such as machines and other equipment.

Practical applications. As mentioned supra, resource loading is part of tactical capacity management, which deals with the development of a rough sketch of the resource usage and timing of the work packages of a given portfolio of orders. Dependent on the application, this may entail due-date setting and accept/reject decisions for individual orders as well as choices regarding overtime and subcontracting. One specific application pertains to the planning of aircraft line maintenance, the routine maintenance between flight arrival and departure at an airport. Line maintenance takes place at the tarmac (the aircraft does not need to be parked in a hangar); typical tasks performed include troubleshooting, minor repairs, systems servicing, and the removal and replacement of components such as batteries, filters, reflectors, ... For each flight, the maintenance must be performed between the scheduled time of arrival and the scheduled time of departure of the aircraft, and the expected maintenance workload is known, see Beliën et al. (2012). Similar applications have been described for maintenance of civil helicopters (Masmoudi, 2011) and for maintenance and repair of national-defence marine equipment (De Boer, 1998). As an application in MTO environments rather than service operations, we mention a tactical production-planning problem in sheet-metal production: each order here consists of one or more jobs that are related via chain-like precedence constraints; and each job visits one specific machine group (a preliminary study was done in the

Master's thesis of Smeulders, 2011).

Contributions. We define three fundamental problem variants. We first consider the decision problem regarding the existence of a feasible schedule that executes all orders with the available workers. Second, we study the scheduling problem of finding a feasible schedule that minimizes the number of additional worker-periods needed to execute all the orders within the planning horizon. Finally, we also look into the combined selection and scheduling problem of choosing a subset of orders that can be executed with the available workers to maximize the total revenue. We identify six special cases that apply to each of these three related problems and we present algorithms for the cases that can be solved either in polynomial time or in pseudo-polynomial time. We prove that the remaining cases are NP-complete in the strong sense and we also discuss the existence of approximation algorithms for some of these cases. Finally, we comment on the validity of our results when orders must be executed without preemption: often indeed, management prefers a schedule in which jobs are not interrupted and resumed at a later time. Although inspired by a number of practical applications, this work focuses on the properties of the underpinning generic combinatorial problems. Our findings contribute to a better understanding of these problems and can also serve as a reference work for authors looking to design efficient algorithms for similar problems.

Organization. The remainder of this article is organized as follows. In Section 2, we give a formal definition of the three problem variants and we review the relevant literature. In Section 3 we study the decision-problem variant, in Section 4 we analyze the scheduling variant, and we investigate resource loading with job selection in Section 5. We conclude in Section 6.

2. Problem statement

We provide a formal problem statement in Section 2.1, we briefly review the relevant literature in Section 2.2, and in Section 2.3 we summarize the results obtained in the following sections.

2.1 Problem description

We consider H consecutive time periods that constitute the planning horizon, where period t ($t = 1, \ldots, H$) runs from time (instant) $t-1$ to t (period 1, for instance, is the interval $[0, 1]$). For aircraft line maintenance, for instance, the planning horizon is typically one week: the same flights are to be serviced every week. In period t, c_t workers are available (this number can be derived from the workforce roster). We denote by $N = \{1, 2, \ldots, n\}$ the set of available jobs (orders), which can be executed by a single resource type (representing the workers) within the planning horizon. Each job j has a release time r_j before which it cannot be started and a deadline $\bar{d}_j$, which represents the latest completion time of the job. In other words, each job j is associated with a time window $[r_j, \bar{d}_j]$. The work content (workload) of job j is denoted by p_j, and is expressed in worker-periods. Furthermore, each job j has a pre-specified minimum number of workers LB_j that must be assigned to that job, and a maximum number of workers UB_j that can be committed to that job, during each period in which it is scheduled ($1 \leq \text{LB}_j \leq \text{UB}_j \leq p_j$). In aircraft maintenance, for example, the size of the aircraft and the specific character of each task impose an upper bound on the number of workers that can work simultaneously and (for safety and coordination) a minimum number of workers should also be present around the aircraft whenever maintenance is performed.

For each job j, we assume that $\text{LB}_j \leq c_t$ for each period t during which j can be executed, and $\text{UB}_j \leq \max\{c_t : t = r_j + 1, \ldots, \bar{d}_j\}$. The decision problem RLP-DECISION answers the question

whether a schedule exists that executes all the jobs within the time horizon while respecting the time windows and the bound and capacity constraints. We denote by RLP-SCHEDULING the problem of finding a schedule that executes all the jobs using the minimum number of extra worker-periods. In practice, this implies the assumption that when needed, external workers will be hired on a per-period basis or that part of the work is subcontracted. Note that by solving RLP-SCHEDULING we can infer the answer to RLP-DECISION. Finally, each job j also has a revenue Q_j, which is collected only when the job is executed, dependent on the job characteristics and on the fee the client is willing to pay. The objective of the third problem, RLP-SELECTION, is (1) to evaluate which jobs should be accepted in order to maximize the total revenue and (2) to derive a feasible schedule that executes all the selected jobs within the time horizon using the available workers. Clearly, RLP-SELECTION is also at least as hard as the problem RLP-DECISION, but no such evident relationship exists between RLP-SCHEDULING and RLP-SELECTION.

A mixed-integer programming (MIP) formulation of these three problem variants uses the binary decision variables x_{jt} defined for $j \in N$ and $t \in \{r_j+1, \ldots, \bar{d}_j\}$, with the interpretation that $x_{jt} = 1$ if and only if job j is executed in period t, and the integer variables $y_{jt} \in \{0, \text{LB}_j, \text{LB}_j + 1, \ldots, \text{UB}_j\}$ ($j \in N$ and $t \in \{r_j + 1, \ldots, \bar{d}_j\}$) representing the number of workers assigned to job j in period t. For RLP-SCHEDULING we also use integer variables z_t ($t = 1, \ldots, H$) that indicate the number of external workers hired in period t. Finally, problem RLP-SELECTION also gives rise to binary variables s_j for $j \in N$, with the interpretation that $s_j = 1$ if and only if job j is selected. A formulation contains the following set of constraints, which says that each job (each selected job, for RLP-SELECTION) should be entirely executed between its release time and its deadline.

$$\sum_{t=r_j+1}^{\bar{d}_j} y_{jt} = p_j s_j, \qquad j = 1, \ldots, n. \tag{1}$$

For RLP-DECISION and RLP-SCHEDULING, all the variables s_j in Equation (1) take the value 1. The workload assigned to each period cannot exceed the number of workers available (including external workers), which translates into:

$$\sum_{j=1}^{n} y_{jt} \leq c_t + z_t, \qquad t = 1, \ldots, H. \tag{2}$$

For RLP-DECISION and RLP-SELECTION, all the variables z_t in Equation (2) are set to 0. Equation (3) ensures that the number of workers assigned to each job in each period is either 0 or within the specified lower and upper bounds.

$$\text{LB}_j \; x_{jt} \leq y_{jt} \leq \text{UB}_j \; x_{jt}, \qquad j = 1, \ldots, n, \quad t = r_j + 1, \ldots, \bar{d}_j. \tag{3}$$

The problem RLP-DECISION corresponds with deciding whether the domain defined by the constraints (1)–(3), with all $s_j = 1$ and $z_t = 0$, is non-empty. Problem RLP-SCHEDULING consists of optimizing the objective function (4) subject to the constraints (1)–(3), with all $s_j = 1$.

$$\min \quad \sum_{t=1}^{H} z_t \tag{4}$$

Finally, RLP-SELECTION amounts to solving

$$\max \quad \sum_{j=1}^{n} Q_j s_j \tag{5}$$

subject to the constraints (1)–(3).

A solution to RLP-DECISION, RLP-SCHEDULING or RLP-SELECTION is entirely determined by the matrix $Y = (y_{jt})_{j,t}$ for $j = 1, \ldots, n$ and $t = 1, \ldots, H$. Indeed, we can deduce the value of the remaining variables as follows: $x_{jt} = 1$ if and only if $y_{jt} > 0$, $s_j = 1$ if and only if $\sum_{t=r_j+1}^{\bar{d}_j} y_{jt} > 0$, and $z_t = \max\{0, \sum_{j=1}^{n} y_{jt} - c_t\}$. Throughout this paper, we denote a solution to RLP-DECISION, RLP-SCHEDULING and RLP-SELECTION by Y, and we resort to the variables $X = \left(x_{jt}\right)_{j,t}$, $S = \left(s_j\right)_j$ and $Z = \left(z_t\right)_t$ only when necessary.

We next present our first result, which states that the integrality constraint on the variables y_{jt} in the MIP formulations can be relaxed if all the parameters are integers.

Proposition 1. *If all the parameters are integers then any optimal solution to* RLP-DECISION*,* RLP-SCHEDULING *or* RLP-SELECTION *obtained using the MIP formulations without integrality constraint on y_{jt} can be transformed in polynomial time into an equivalent solution satisfying the restriction $y_{jt} \in \{0, \text{LB}_j, \text{LB}_j + 1, \ldots, \text{UB}_j\}$ for all $j \in N$ and $t = 1, \ldots, H$.*

Proof: We prove this result for the problem RLP-SELECTION, the reasoning for RLP-DECISION and RLP-SCHEDULING is analogous. Assume that all parameters are integers and let Y' be an optimal solution to the MIP formulation for RLP-SELECTION without integrality constraint (so $y_{jt} \geq 0$ for $j \in N$ and $t = 1, \ldots, H$). Suppose that some components of Y' are not integer. If there is a unique job $j_0 \in \{1, \ldots, n\}$ for which some $y'_{j_0 t}$ are fractional then we define an integer Y as follows. Let t_1 be the first period (smallest index) with $y'_{j_0 t_1}$ fractional; we build a new solution Y with integer y_{jt} for $t \leq t_1$ and for all j by defining $y_{jt} = y'_{jt}$ for $j \in N \setminus \{j_0\}$ and $t = 1, \ldots, H$; and $y_{j_0 t} = \lceil y'_{j_0 t} \rceil$ for $t \leq t_1$ (here $\lceil \cdot \rceil$ is the symbol for the ceiling); the remaining $y_{j_0 t}$ with $t > t_1$ are adapted accordingly. More concretely, let $t_2 > t_1$ be the first period such that $y_{j_0 t_1} - y'_{j_0 t_1} \leq \sum_{t=t_1+1}^{t_2} \left(y'_{j_0 t} - \lfloor y'_{j_0 t} \rfloor\right)$ (where $\lfloor \cdot \rfloor$ is the symbol for the floor), then $y_{j_0 t} = \lfloor y'_{j_0 t} \rfloor$ for $t_1 < t < t_2$; $y_{j_0 t_2} = \lfloor y'_{j_0 t_2} \rfloor + \left[\sum_{t=t_1+1}^{t_2} \left(y'_{j_0 t} - \lfloor y'_{j_0 t} \rfloor\right) - y_{j_0 t_1} + y'_{j_0 t_1}\right]$ and $y_{j_0 t} = y'_{j_0 t}$ for $t > t_2$. By repeating this procedure at most H times, we arrive at a solution Y whose components are all integer. Furthermore, Y is a solution to RLP-SELECTION that executes the same jobs as Y', and hence Y and Y' have the same objective value.

Now suppose that more than one job has fractional components. Without loss of generality, we assume that for each $t = 1, \ldots, H$, either $y'_{jt} \in \{0, \text{LB}_j, \text{LB}_j + 1, \ldots, \text{UB}_j\}$ for all $j \in N$ or there are at least two jobs j_1 and j_2 that are executed with a fractional number of workers in that period. If this is not the case, we simply consider the equivalent solution where for that unique job, say j_1, we have replaced $y'_{j_1 t}$ by $\lceil y'_{j_1 t} \rceil$, and the execution of job j_1 in the subsequent periods is adapted as described above. Using a similar argument, we can assume that for each period t the quantity $\sum_{j=1}^{n} y'_{jt}$ is integer.

Let Y' be an optimal solution to the relaxation of the MIP formulation for RLP-SELECTION satisfying the above restrictions. We build the following weighted complete bipartite graph $G = (V_1 \cup V_2, E)$, where each node in V_1 corresponds with a job j having at least one period t where y'_{jt} is fractional and each node in V_2 corresponds with a period t with at least one job j with y'_{jt} fractional; we write $V_1 = \{j_1, j_2, \ldots, j_\alpha\}$ and $V_2 = \{t_1, t_2, \ldots, t_\beta\}$, where $\alpha \leq n$ and $\beta \leq H$. Each node $i \in V_1$ is a source node with supply $\sum_{t=r_i+1}^{\bar{d}_i} \{y'_{it} - \lfloor y'_{it} \rfloor\}$, and each node $t \in V_2$ is a sink node with demand $\sum_{j=1}^{n} \left\{y'_{jt} - \lfloor y'_{jt} \rfloor\right\}$. Finally, each arc $(i, t) \in V_1 \times V_2$ has unit capacity, and zero cost if $y'_{it} \neq \lfloor y'_{it} \rfloor$ and cost $= M$ otherwise, where M is a large positive number. All the arc capacities are integer, so the *Integrality Theorem* (Ahuja et al., 1993) implies that there exists an optimal integer solution to the resulting capacitated network flow instance. Furthermore, since there is a

fractional solution with cost 0, we infer that any optimal integer solution also has objective value 0. Let $A = (a_{j_k t_\ell})_{k,\ell}$, for $k = 1, \ldots, \alpha$ and $\ell = 1, \ldots, \beta$, be such an optimal integer solution, which can obtained in polynomial time (Ahuja et al., 1993). We consider the matrix Y defined as follows: for $(j,t) \notin V_1 \times V_2$ we have $y_{jt} = y'_{jt}$ and for $(j,t) \in V_1 \times V_2$ we set $y_{jt} = \lfloor y'_{jt} \rfloor + a_{jt}$. Matrix Y is an integer solution to RLP-SELECTION that executes the same jobs as Y', therefore Y and Y' have the same objective value. □

We close this section with additional definitions. We say that a job $j \in N$ is *splittable* if and only if $2\,\text{LB}_j \leq p_j$ and $r_j + 1 < \bar{d}_j$; these are indeed two necessary conditions for a job to be executable over more than one period. Note that a splittable job need not be executed in multiple periods in an optimal schedule. A job $j \in N$ is said to be executed *without preemption* if and only if it is executed in consecutive periods, meaning that all periods with $y_{jt} \geq 1$ are consecutive, for all $j \in N$. When $\text{LB}_j \geq 1$ for $j = 1, \ldots, n$, Proposition 1 remains valid even if each job must be executed without preemption; in Section 2.3 we will conclude that case $\text{LB}_j = 0$ can be replaced by $\text{LB}_j = 1$, and so the result holds regardless of the values of the lower bounds.

2.2 Literature review

Resource loading. This paper models tactical capacity planning in equipment maintenance, MTO and ETO; there are few studies dealing with similar problems. The articles most related to our work are those that focus on tactical planning and use a "bucketized" time horizon (each period is a "bucket"). This includes Hans (2001) and Mestry et al. (2011); both references develop a branch-and-price algorithm. Variants of our problem are studied by Wullink et al. (2004), whose major contribution is the incorporation of uncertainty, Gademann and Schutten (2005), who propose linear-programming-based heuristics, and Alfieri et al. (2011), who study the inclusion of "feeding" precedence constraints, which allow some overlap in the execution of precedence-related activities. The use of a discrete time horizon is quite common in medium-term production planning for make-to-stock environments, but the underlying models are otherwise quite different (these models generally formalize lot-sizing decisions; see Pochet and Wolsey, 2006, for instance). In lot-sizing terminology, we are working with "big buckets" rather than "small buckets," because more than one job can be scheduled in the same bucket (see Suerie and Stadtler, 2003); in our model, we will decide *which proportion* of the workload of each job to perform during each time period.

In the operational planning literature, models quite similar to our planning of work packages have been studied under the title of project scheduling with "variable-intensity" or "continuously divisible" activities; examples are Leachman (1983); Leachman et al. (1990); Kogan and Shtub (1999). The first two papers only describe heuristic algorithms, while the last one provides a characterization of optimal solutions and preliminary computational results. One of the most recent contributions to this body of literature is Kis (2005), who reports computational results for benchmark data sets based on a branch-and-cut algorithm. An early reference, which considers a continuous time horizon, is Weglarz (1981). A different line of literature adheres more closely to operational machine scheduling, such as Sadykov (2012); Kim et al. (2004); Serafini (1996); Xing and Zhang (2000), and Günther et al. (2014); Hendel et al. (2014); Jansen and Porkolab (2002); Jansen (2004); Mounié et al. (2007) who investigate the problem of scheduling "malleable jobs," whereas Baptiste et al. (1999) focus on the "fully elastic cumulative scheduling problem," with an equivalent solution space. The models studied in the above references differ from the variants of resource loading investigated in this paper; in particular, to the best of our knowledge, apart from Baptiste et al. (1999) none of the theoretical work cited has considered time windows, lower and upper bounds, or job selection; the reference Baptiste et al. (1999) does consider time windows and contains a result that corresponds to our Lemma 2. In Section 5.5 of his recent handbook,

Drozdowski (2009) provides a very thorough summary of parallel-machine scheduling with malleable jobs, with some pointers to isolated results in multiprocessor scheduling where also upper bounds or ready times are considered, but the problem statement never fully coincides with ours.

Order selection. In today's competitive manufacturing environment, organizations commit to meeting order deadlines agreed to with customers. However, order acceptance often takes place without consideration of the effect on the planning of the other jobs in the order portfolio. As pointed out by Herbots et al. (2007) and Guerrero and Kern (1998), an integration of job selection and planning is essential within a firm to avoid conflicts of interest between the sales department, which tends to accept as many jobs as possible, and the production department, which attempts to meet the promised delivery dates to clients.

Within the domain of tactical job-shop planning, the integration of order acceptance decisions has already been investigated; for some recent work, see Ebben et al. (2005); Slotnick and Morton (2007). Excellent literature surveys on the topic of order acceptance and scheduling are provided by Keskinocak and Tayur (2004); Roundy et al. (2005) and Slotnick (2011). In a more classic scheduling context, the goal of "maximum-throughput scheduling" is to find a non-preemptive schedule that maximizes the weight of the jobs that meet their deadline (see, for instance, Bar-Noy et al., 2001; Engels et al., 2003). In the standard notation for scheduling problems, this boils down to variants of $P|r_i|\sum(1-U_i)$, a study which goes back at least to Moore (1968).

2.3 Our results

We consider each of the problems RLP-DECISION, RLP-SCHEDULING and RLP-SELECTION separately in Sections 3, 4 and 5, respectively. In each section, we focus on the the following special cases: (1) no time windows and no bound constraints, (2) no bound constraints, (3) no lower bounds, (4) no upper bounds, (5) there is only one job, and finally (6) there are exactly two jobs. In this

Table 1: Summary of our results: 'ordinary' means NP-complete/hard in the ordinary sense, 'strong' means NP-complete/hard in the strong sense.

	Preemption	RLP-DECISION	RLP-SCHEDULING	RLP-SELECTION
No constraints	yes	$O(n)$	$O(n)$	ordinary
	no	$O(n)$	$O(n)$	ordinary
No bound constraints	yes	$O(n \ln n)$	$O(n \ln n)$	ordinary
	no	strong	strong	strong
No lower bounds	yes	$O(n \ln n)$	$O(n \ln n)$	ordinary
	no	strong	strong	strong
No upper bounds	yes	strong	strong	strong
	no	strong	strong	strong
One job	yes	$O(H \ln H)$	$O(H \ln H)$	$O(H \ln H)$
	no	$O(H)$	$O(H)$	$O(H)$
Two jobs	yes	ordinary	ordinary	ordinary
	no	$O(H^4)$	$O(H^4)$	$O(H^4)$

context, the case 'no time windows' corresponds with the same time window $[0, H]$ for each job; 'no upper bound' equates with $UB_j = p_j$, and 'no lower bound' means that $LB_j = 0$, which is equivalent with $LB_j = 1$ because the y_{jt}-values are integer and the lower bound only applies for periods t where $y_{jt} > 0$. The cases with one and two jobs are interesting because they may be encountered as subproblems while solving problems with higher n; our algorithm for the decision problem with one job, for instance, is also called as a subroutine to decide the case with two jobs (see Appendices A and B). The results of our analysis are summarized in Table 1. Unless otherwise mentioned, in cases (1) to (4) the parameter H is considered to be constant, since typically $n > H$, whereas for cases (5) and (6) we have only one or two jobs and H is then regarded as input.

3. Problem RLP-DECISION

We consider the problem RLP-DECISION and we study the six special cases presented in the previous section. For ease of exposition, we alternatively see each job $j \in N$ as consisting of p_j *joblets*, where a joblet corresponds with the work performed by one worker in one period. Similarly, the available capacity in period t can be divided into c_t units, where each unit equates with one worker.

3.1 No time windows and no bound constraints

All jobs have the same time window $[0, H]$ and any number of available workers can be assigned to any job. We observe the following:

Lemma 1. *A given instance of* RLP-DECISION *with the above restrictions is a* Yes *instance if and only if* $\sum_{j=1}^{n} p_j \leq \sum_{t=1}^{H} c_t$.

Proof: Each joblet (of any job) can be assigned to any worker in any period; such assignment is feasible if and only if there is enough capacity. This condition is exactly the inequality $\sum_{j=1}^{n} p_j \leq \sum_{t=1}^{H} c_t$. □

Lemma 1 remains valid even if each job must be executed without preemption.

3.2 No bound constraints

We now assume that jobs can have different time windows but there are no bound constraints, so the resulting problem is a generalization of the problem defined in Section 3.1. We propose Algorithm 1 for solving RLP-DECISION when there are no bound constraints.

Algorithm 1 RLP-DECISION without bound constraints

1: compute an ordering π of the jobs in non-decreasing order of their deadline
2: **for** $j = 1$ to n **do**
3: schedule the joblets of job $\pi[j]$ starting from period $r_{\pi[j]} + 1$;
when a period is full, move to the next period
4: **if** all the jobs can be executed **then** return Yes
5: **else** return No

Algorithm 1 is a modification of the *earliest due-date* rule (Pinedo, 2008), allowing to schedule a fraction of a job in one period, if necessary.

Lemma 2. *Algorithm 1 solves* RLP-DECISION *without bound constraints in time* $O(n \ln n)$.

Proof: For ease of exposition, we assume that jobs are already indexed in non-decreasing order of their deadline; this can be done in $O(n \ln n)$, where n is the number of jobs (Cormen et al., 2005).

On the one hand, if Algorithm 1 outputs YES then the computed schedule is such that each job is executed within its time window and the capacity of each period is respected; in other words, all the constraints of RLP-DECISION are satisfied. On the other hand, suppose that we have a YES instance of RLP-DECISION without bound constraints and that we are given a feasible schedule. Consider the first job (with the smallest deadline) and its release time r_1. Move all the joblets of job 1 to period $r_1 + 1$ and, if that period is not enough to completely schedule job 1 then use the next period. Repeat this until job 1 is completely executed. Note that the obtained schedule is feasible because all the jobs (or joblets) that are moved backward have their deadline greater than or equal to that of job 1. This procedure is successively repeated for jobs $2, \ldots, n$, without moving jobs (or joblets of jobs) with smaller indices that are already re-scheduled. The final schedule is feasible because we started with a feasible schedule. Furthermore, the latter is exactly the schedule that is output by Algorithm 1. As a consequence, Algorithm 1 returns YES for this instance.

The time complexity of Algorithm 1 is dominated by the sorting in line 1; therefore its complexity is $O(n \ln n)$. □

Baptiste et al. (1999) obtain a comparable result based on different reasoning. Algorithm 1 does not guarantee the execution of jobs without preemption, as illustrated by the following example. Consider an instance of RLP-DECISION without bound constraints consisting of two jobs that must be executed in five periods ($H = 5$), each with capacity $c_t = 5$ workers (for $t = 1, \ldots, 5$). Job 1 has $r_1 = 0$, $\bar{d}_1 = 5$ and $p_1 = 17$, and job 2 is characterized by $r_2 = 1$, $\bar{d}_2 = 3$ and $p_2 = 8$. Algorithm 1 schedules job 2 in periods 2–3, with five and three workers respectively, and executes job 1 in periods 1, 3, 4 and 5, with five, two, five and five workers, respectively. Job 1 is preempted because it is executed in period 1 and 3 but not in period 2. A schedule that executes both jobs without preemption does exist, however: it executes job 2 in periods 2–3 with four workers in each period.

The next result shows that it is unlikely that there exists a polynomial-time algorithm for solving RLP-DECISION without bound constraints when each job must be executed without preemption. In fact, we show that if preemption is not allowed then the problem is NP-complete in the strong sense. We prove this result using a reduction from SEQUENCING WITH RELEASE TIMES AND DEADLINES, which is NP-complete in the strong sense (Garey and Johnson, 1979, page 236). The latter problem is defined as follows:

SEQUENCING WITH RELEASE TIMES AND DEADLINES
Instance: Set T of tasks and for each task $t \in T$, a length $\ell(t) \in \mathbb{Z}^+$, a release time $r(t) \in \mathbb{Z}_0^+$, and a deadline $d(t) \in \mathbb{Z}^+$.
Question: Is there a one-processor schedule for T that satisfies the release-time constraints and meets all the deadlines, i.e., a one-to-one function $\sigma : T \to \mathbb{Z}_0^+$, with $\sigma(t) > \sigma(t')$ implying $\sigma(t) \geq \sigma(t') + \ell(t')$, such that, for all tasks $t \in T$, $\sigma(t) \geq r(t)$ and $\sigma(t) + \ell(t) \leq d(t)$?

We state the following complexity result.

Lemma 3. *The problem* RLP-DECISION *without bound constraints and where each job must be executed without preemption is* NP*-complete in the strong sense.*

Proof: Clearly, the problem RLP-DECISION without bound constraints belongs to the class NP. Given an arbitrary instance of SEQUENCING WITH RELEASE TIMES AND DEADLINES, we construct an instance of RLP-DECISION with $H = \max_{t \in T} d(t)$ periods and each period has exactly one worker. There are $|T|$ jobs and each job corresponds with a task in T such that job j has processing time $p_j = \ell(j)$ and time window $[r_j, \bar{d}_j] = [r(j), d(j)]$. Note that this construction can be done

in polynomial time. Because each job must be executed without preemption and each period has exactly one worker, it is immediate that there exists a feasible schedule to the instance of SEQUENCING WITH RELEASE TIMES AND DEADLINES if and only if the constructed instance of RLP-DECISION without bound constraints is a YES instance. □

3.3 No lower bounds

The execution of each job is constrained only by its time window and the upper bound. Algorithm 1 can be modified to handle also this situation, if in line 3 the capacity of each period t is filled up with $\min\left\{\text{UB}_{j_{\pi[j]}}, p_{\pi[j]}, c_t\right\}$ before moving to the next period.

Again, the resulting solution does not necessarily schedule jobs without preemption. When the latter restriction is imposed, the proof of Lemma 3 remains valid to show that RLP-DECISION without lower bounds and without preemption is NP-complete. We formalize this result in the next lemma.

Lemma 4. *The problem* RLP-DECISION *without lower bounds and where each job must be executed without preemption is* NP*-complete in the strong sense.*

3.4 No upper bounds

This is the case where there are no upper-bound restrictions (only the available number of workers in each period). The problem RLP-DECISION with no upper bounds is a special case of the maximum flow problem with minimum quantities (Thielen and Westphal, 2012; Haugland et al., 2011), because the associated network is bipartite. As a consequence, any positive result for the latter problem remains valid, but the negative (results do not necessarily apply to RLP-DECISION without upper bounds.

RLP-DECISION without upper bounds can be solved using Lemma 1 if it consists of a single period. The next result states that the problem becomes NP-complete as soon as there are at least two periods. The proof uses a reduction from the following variant of PARTITION, which easily reduces from the variant shown to be NP-complete by Garey and Johnson (1979).

PARTITION
Instance: A set A with $2m$ elements, and a size $s(a) \in \mathbb{Z}^+$ with $s(a) > 2$ for each $a \in A$ such that $\sum_{a\in A} s(a) = 2B$.
Question: Can A be partitioned into two disjoint sets A_1 and A_2 such that $|A_1| = |A_2| = m$, and $\sum_{a\in A_1} s(a) = B$?

Lemma 5. *The problem* RLP-DECISION *with no upper bounds is* NP*-complete, even if there are two periods and all jobs are splittable.*

Proof: The problem RLP-DECISION with no upper bounds clearly belongs to the class NP. Given an arbitrary instance of PARTITION, we build an instance of RLP-DECISION with two periods ($H = 2$). There are $c_1 = mB^3 + B^2 - m$ workers available in the first period and $c_2 = 3mB^3 + 3B^2 - m$ workers in the second one. There are $2m$ jobs, and each job corresponds with an element in A such that job j has the processing time $p_j = 2B^3 + 2s(j)B - 1$ and $\text{LB}_j = B^3 + s(j)B - 1$. Furthermore, all the jobs have the same time window $[0, 2]$. This construction can be done in polynomial time. Note that all the jobs are splittable, and that there is only one way to split job j, namely into a smaller part requiring $p_{j_s} = B^3 + s(j)B - 1$ workers, which we refer to as j_s, and a larger part j_ℓ with $p_{j_\ell} = B^3 + s(j)B$ workers. We now argue that we have a YES instance of PARTITION if and only if the instance of RLP-DECISION is also a YES instance.

On the one hand, if we have a YES instance of PARTITION then we construct a feasible schedule as follows: all the jobs corresponding with elements in A_1 are split into two parts. The smaller parts are scheduled in the first period whereas the larger parts of these jobs and the jobs corresponding with elements in A_2 are all scheduled in the second period. It can be verified that this schedule is feasible for the instance of RLP-DECISION. On the other hand, suppose that we have a YES instance of RLP-DECISION. The number of workers required to execute all the jobs is equal to the sum of the capacities of the two periods, so the capacity of each period is fully used. Let us consider a feasible schedule and denote by F the set of jobs fully executed in the first period and by S (respectively L) the set of split jobs whose smaller (respectively larger) part is executed in the first period. The capacity usage (and availability) in period 1 is then:

$$2|F|B^3 + 2B\sum_{j\in F} s(j) - |F| + |S|B^3 + B\sum_{j\in S} s(j) - |S| + |L|B^3 + B\sum_{j\in L} s(j) = mB^3 + B^2 - m. \quad (6)$$

Without loss generality, we assume that B is sufficiently larger than m such that the coefficients of B^3 and the constant terms in both sides of Equation (6) coincide. It follows that

$$2|F| + |S| + |L| = m, \text{ and} \quad (7a)$$
$$|F| + |S| = m. \quad (7b)$$

By substituting (7b) into (7a), we obtain that $|F| + |L| = 0$, which implies that $|F| = |L| = 0$ and $|S| = m$. This means that exactly m smaller parts of split jobs are executed in the first period. By comparing the coefficients of B^2 in Equation (6) we obtain that $\sum_{j\in S} s(j) = B$. We then define A_1 as follows: an element $a \in A_1$ if and only if the smaller part of the corresponding job is scheduled in the first period. The remaining m jobs (that are not split) belong to A_2. Clearly, we have $|A_1| = |A_2| = m$. Furthermore, since $\sum_{a\in A_1} s(a) = B$ we conclude that A_1 and A_2 constitute a feasible partition of A. This completes the proof of Lemma 5. □

Lemma 5 indicates that RLP-DECISION with no upper bounds is at least NP-complete in the ordinary sense, leaving open the possibility that the problem is even harder, namely NP-complete in the strong sense. Below, we present a pseudo-polynomial-time algorithm for solving the problem when there are exactly two periods and we show that the problem becomes effectively NP-complete in the strong sense when there number of periods is free.

Consider RLP-DECISION with two periods and no upper bounds. If $\sum_{j\in N} p_j > c_1 + c_2$ then the answer is NO; otherwise we can call the following dynamic programming (DP) algorithm for a decision. Without loss of generality, we assume that $r_j = 0$ and $\bar{d}_j = 2$ for each job $j \in N$. For ease of exposition, we add a dummy job 0 with $r_0 = 0$, $\bar{d}_0 = 2$ and $p_0 = 0$. We use the following (Boolean) DP function:

$$F_k(\alpha_1, \alpha_2) \equiv \begin{cases} 1, \text{ if jobs } 0, 1, \ldots, k, \text{ can be executed with } \alpha_t \text{ workers in period } t \ (t = 1, 2), \\ 0, \text{ otherwise.} \end{cases}$$

The initial condition of the recursion is given by $F_0(\alpha_1, \alpha_2) = 1$ for all $\alpha_1 \in \{0, 1, \ldots, c_1\}$ and $\alpha_2 \in \{0, 1, \ldots, c_2\}$, and by $F_0(\alpha_1, \alpha_2) = 0$ if $\alpha_1 \in \{-c_1, \ldots, -1\}$ or $\alpha_2 \in \{-c_2, \ldots, -1\}$. The recursive relation is

$$F_k(\alpha_1, \alpha_2) = \begin{cases} 0, \text{ if } \alpha_1 \in \{-c_1, \ldots, -1\} \text{ or } \alpha_2 \in \{-c_2, \ldots, -1\}, \\ \max\{F_{k-1}(\alpha_1 - \ell, \alpha_2 + \ell - p_k) : \ell \in \Delta_k\}, \text{ otherwise,} \end{cases}$$

where $\Delta_k = \{0, p_k\} \cup \left(\mathbb{N} \cap \left[\text{LB}_k, p_k - \text{LB}_k\right]\right)$. If job k is not splittable then the second part of the union is empty, and the job will be executed either exclusively in period 1 or exclusively

in period 2; splittable jobs may also be executed in both periods, dependent on the remaining capacity. The solution to RLP-DECISION with two periods and without upper bounds is obtained by inspecting the value of $F_n(c_1, c_2)$: if $F_n(c_1, c_2) = 1$ then we have a YES instance, otherwise we have a NO instance. The time complexity of this DP algorithm is $O\left(C^2\, n\right)$, where $C = \max\{c_1, c_2\}$. Therefore, RLP-DECISION with two periods can be solved in pseudo-polynomial time. This DP recursion can be generalized for any constant H: the total number of partitions $(\ell_1, \ell_2, \ldots, \ell_H)$ of p_k, with $\sum_{i=1}^{H} \ell_i = p_k$, is then constant in the number of jobs, and therefore the resulting algorithm would still run in pseudo-polynomial time.

When the number of periods is free (H is part of the input), we show that RLP-DECISION without upper bounds is NP-complete in the strong sense. The proof uses a reduction from 3-PARTITION, which is defined as follows (Garey and Johnson, 1979):

3-PARTITION
Instance: A set A of $3m$ elements, a bound $B \in \mathbb{Z}^+$, and a size $s(a) \in \mathbb{Z}^+$ for each $a \in A$ such that $\frac{B}{4} < s(a) < \frac{B}{2}$ and $\sum_{a \in A} s(a) = mB$.
Question: Can A be partitioned into m disjoint sets A_1, A_2, ..., A_m such that, for $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = B$.

Lemma 6. *The problem* RLP-DECISION *with no upper bounds is* NP*-complete in the strong sense, even if all jobs are splittable.*

Proof: Given an arbitrary instance of 3-PARTITION, we construct an instance of RLP-DECISION with $H = m + 1$ periods. The first m periods have $3B^3 + B^2 - 3$ workers available and the last period $(m + 1)$ has $3mB^3 + mB^2$ workers. There are $3m$ jobs, each job corresponding with an element of A such that job j has the processing time of $p_j = 2B^3 + 2s(j)B - 1$ and the lower bound of $\text{LB}_j = B^3 + s(j)B - 1$. All jobs have the same time window $[0, H]$. Similarly as in the proof of Lemma 5, these jobs are all splittable and there is only one way to split each job j, with a smaller part j_s requiring $p_{j_s} = B^3 + s(j)B - 1$ workers and a larger part j_ℓ with $p_{j_\ell} = B^3 + s(j)B$ workers. This construction can be completed in polynomial time. We now argue that the instance of 3-PARTITION is a YES instance if and only if the constructed instance of RLP-DECISION is also a YES instance.

On the one hand, if we have a YES instance of 3-PARTITION then there exists a feasible schedule to the instance RLP-DECISION. Indeed, it suffices to execute in period t the smaller part of the three jobs corresponding with elements in A_t, for $t = 1, \ldots, m$; and then execute all the larger parts of all the jobs in the last period.

On the other hand, let us suppose that we have a feasible schedule to the constructed instance of RLP-DECISION. The number of workers required to execute all the jobs is equal to the sum of the capacities of all the periods, so we infer that the capacity of each period is fully used. We denote by F the set of jobs completely executed in the last period $(m + 1)$, and by S (respectively L) the set of jobs for which only the smaller (respectively larger) part is executed in period $(m + 1)$. The capacity usage in period $(m + 1)$ is then:

$$2|F|B^3 + 2B\sum_{j \in F} s(j) - |F| + |S|B^3 + B\sum_{j \in S} s(j) - |S| + |L|B^3 + B\sum_{j \in L} s(j) = 3mB^3 + mB^2. \quad (8)$$

It follows that

$$2|F| + |S| + |L| = 3m, \text{ and} \quad (9a)$$
$$|F| + |S| = 0; \quad (9b)$$

this implies that $|F| = |S| = 0$ and $|L| = 3m$. This observation indicates that all the jobs are split (into two parts), and that the larger parts are all scheduled in period $m + 1$ whereas the smaller parts are all executed in the first m periods. Each of these first m period is then assigned exactly three smaller parts, which fully occupy the available workers. We now construct a partition of A by defining A_ℓ ($\ell = 1, \ldots, m$) to contain the three elements for which the smaller parts of their corresponding jobs are scheduled in period ℓ. This completes the proof of Lemma 6. □

We close this section by making the following two observations. First, all the results presented in this section remain valid when all jobs have the same time window. Second, the DP algorithm developed for instances with two (or more, but constant number of) periods, as well as Lemma 5, remain valid even if each job must be executed without preemption. The negative result in Lemma 3 also directly transfers, which leads us to conclude that RLP-DECISION without upper bounds is NP-complete in the strong sense even if each job must be executed without preemption.

3.5 There is only one job

We present Algorithm 2 for solving RLP-DECISION with a single job. In the description we omit the subscript referring to the job. Without loss of generality, we assume that the release date is $r = 0$ and the deadline $\bar{d} = H$.

Algorithm 2 RLP-DECISION with a single job

1: sort periods in non-increasing order of the available number of workers
2: **if** $p > \sum_{t=1}^{H} \min\{c_t, \text{UB}\}$ **then** return NO
3: **if** $\text{LB} \cdot H \leq p$ **then** return YES
4: **else**
5: find the largest integer H' such that $\text{LB} \cdot H' \leq p$
6: **if** $p \leq \sum_{t=1}^{H'} \min\{c_t, \text{UB}\}$ **then** return YES
7: **else** return NO

Lemma 7. *Algorithm 2 solves* RLP-DECISION *with a single job in time* $O(H \ln H)$.

In Appendix A, we present a modified version of Algorithm 2 that solves RLP-DECISION with a single job that must be executed without preemption in time $O(H)$.

3.6 There are two jobs

We establish the following complexity result for instances of RLP-DECISION that consist of two jobs.

Lemma 8. RLP-DECISION *with two jobs is* NP*-complete, even if the two jobs are identical.*

Proof: We prove this result using a reduction from the problem PARTITION defined earlier. Given an arbitrary instance of PARTITION, we build an instance of RLP-DECISION with two identical jobs ($n = 2$) as follows. Each job j ($j = 1, 2$) has the workload $p_j = (2m+1)B$, $\text{LB}_j = 2B + \min_{a \in A} s(a)$ and $\text{UB}_j = 2B + \max_{a \in A} s(a)$. There are $2m$ periods with $c_t = 2B + s(t)$ workers available in period t, for $t = 1, \ldots, 2m$. The lower bound implies that in each period we can execute only one job. Finally, the time window of each job is $[0, 2m]$. This construction can be completed in polynomial time. We now argue that the instance of PARTITION is a YES instance if and only if the instance of RLP-DECISION is also a YES instance.

On the one hand, if we have a YES instance of PARTITION then the set A can be partitioned into two subsets A_1 and A_2 such that $\sum_{a \in A_1} s(a) = B$. For each period t, if $t \in A_1$ then we assign $2B + s(t)$ workers to job 1 and 0 workers to job 2 in that period; otherwise, we assign $2B + s(t)$ workers to job 2 and 0 workers to job 1 in that period. The schedule leads to a feasible execution of both jobs and we conclude that the instance of RLP-DECISION is a YES instance. On the other hand, if we have a YES instance of RLP-DECISION then by defining $A_i = \{t : \text{job } i \text{ is executed in period } t\}$, we arrive at the conclusion that the instance of PARTITION is a YES instance. □

Similarly as for Lemma 5, the result of Lemma 8 also leaves open the possibility that the problem is NP-complete in the strong sense. We discard this option by presenting a pseudo-polynomial-time DP algorithm for solving instances of RLP-DECISION that consist of two jobs. Note that Algorithm 2 can be used for each job individually to identify possible NO instances. For ease of exposition, we present a DP algorithm for the setting where the two jobs have the same time window. We define the Boolean value function

$$F_t(\beta_1, \beta_2) \equiv \begin{cases} 1, & \text{if a schedule exists that assigns exactly } \beta_i \text{ workers in total to job } i \ (i = 1, 2) \\ & \quad \text{during periods } 1, \ldots, t, \\ 0, & \text{otherwise.} \end{cases}$$

The initial condition is $F_1(\beta_1, \beta_2) = 1$ if $\beta_1 + \beta_2 \leq c_1$ and $\beta_j \in \{0, \text{LB}_j, \text{LB}_j + 1, \ldots, \text{UB}_j\}$ for $j = 1, 2$; and $F_1(\beta_1, \beta_2) = 0$, otherwise. The recursive relation is

$$F_t(\beta_1, \beta_2) = \max\Big\{F_{t-1}(\beta_1 - \ell_1, \beta_2 - \ell_2) : \ell_j \in \{0, \text{LB}_j, \text{LB}_j + 1, \ldots, \text{UB}_j\}, \text{ for } j = 1, 2\Big\}.$$

An optimal solution to an instance of RLP with two jobs is YES if $F_H(p_1, p_2) = 1$ and NO otherwise. The time complexity of this DP algorithm is $O\left(P^2 H\right)$, where $P = \max\{p_1, p_2\}$.

We close this section by observing that RLP-DECISION with two jobs becomes polynomially solvable when the two jobs must be executed without preemption; in Appendix B we present an algorithm that runs in time $O(H^4)$ for solving this problem.

4. Problem RLP-SCHEDULING

We now turn our attention to the problem RLP-SCHEDULING. As mentioned earlier, by solving RLP-SCHEDULING we can infer the solution to RLP-DECISION, which implies that all the negative results established in Section 3 are also valid for RLP-SCHEDULING under the same conditions.

4.1 No time windows and no bound constraints

We have the following result, which is an adaptation of Lemma 1.

Lemma 9. *For a given instance of* RLP-SCHEDULING *with the above restrictions, the quantity* $\max\left\{0, \sum_{j=1}^{n} p_j - \sum_{t=1}^{H} c_t\right\}$ *represents the optimal objective value.*

A schedule that achieves the objective value of $\max\left\{0, \sum_{j=1}^{n} p_j - \sum_{t=1}^{H} c_t\right\}$ is obtained by executing jobs (in any order) from the first period to the last-but-one period without external workers, and in the last period all the remaining jobs are scheduled, which then requires $\max\left\{0, \sum_{j=1}^{n} p_j - \sum_{t=1}^{H} c_t\right\}$ external workers. The lemma remains valid even with a non-preemption constraint.

4.2 No bound constraints

We present Algorithm 3, which is an adaptation of Algorithm 1, for solving RLP-SCHEDULING when there are no bound constraints.

Algorithm 3 RLP-SCHEDULING without bound constraints

1: $z_t = 0$ for $t = 1, \ldots, H$
2: compute π as an ordering of the jobs in non-decreasing order of their deadline
3: **for** $j = 1$ to n **do**
4: schedule the joblets of job $\pi[j]$ starting from period $r_{\pi[j]} + 1$ as described by Algorithm 1
5: **if** period $\bar{d}_{\pi[j]}$ is full **then** add to $z_{\bar{d}_{\pi[j]}}$ the number of unscheduled joblets of job j
6: return $\sum_{t=1}^{H} z_t$

A proof of the following result can follow similar reasoning as the proof of Lemma 2.

Lemma 10. *Algorithm 3 solves the problem* RLP-SCHEDULING *without bound constraints in time* $O(n \ln n)$.

In the same way as Algorithm 1, Algorithm 3 also does not guarantee the execution of jobs without preemption. When this restriction is imposed, the problem RLP-SCHEDULING without bound constraints becomes NP-hard in the strong sense, as a consequence of Lemma 3. We further establish the following non-approximability result.

Lemma 11. *Unless* P = NP*, problem* RLP-SCHEDULING *does not admit an approximation algorithm when there are no bound constraints and with a non-preemption constraint.*

Proof: Let us consider the reduction from SEQUENCING WITH RELEASE TIMES AND DEADLINES constructed in the proof of Lemma 3. We have a YES instance of SEQUENCING WITH RELEASE TIMES AND DEADLINES if and only if the constructed instance of RLP-SCHEDULING has an optimal objective value of 0. As a consequence, if there exists an approximation algorithm for the latter problem then it can be used to solve the former problem. In combination with the fact that SEQUENCING WITH RELEASE TIMES AND DEADLINES is NP-complete, the lemma can then be seen to hold. □

4.3 No lower bounds

Analogously to Section 3.3, we observe that Algorithm 3 can be modified to incorporate non-trivial upper bounds. We conclude here that both for the decision variant as well as for the scheduling problem, the complexity status without lower bounds is essentially the same as without bound constraints, which leads to the conclusion that upper bounds are easier to deal with than lower bounds (see below), at least without preemption.

We can also borrow the following two results from Section 4.2: the problem RLP-SCHEDULING without lower bounds and with non-preemption constraint is NP-hard in the strong sense, and does not admit an approximation algorithm unless P = NP.

4.4 No upper bounds

Lemma 5 implies that the problem RLP-SCHEDULING without upper bounds is NP-hard, even if there are only two periods and all jobs are splittable. We now present a DP algorithm for solving instances of RLP-SCHEDULING with no upper bounds when there are exactly two periods ($H = 2$). We assume that $r_j = 0$ and $\bar{d}_j = 2$ for each job $j \in N$ and we add a dummy job 0 with $r_0 = 0$,

$\bar{d}_0 = 2$ and $p_0 = 0$. We use the value function $F_k(\alpha_1, \alpha_2)$, which represents the minimum number of additional worker-periods necessary to execute jobs $0, 1, \ldots, k$ with α_t workers in period t ($t = 1, 2$). The initial condition of the recursion is given by $F_0(\alpha_1, \alpha_2) = 0$ for all $\alpha_1 \in \{0, 1, \ldots, c_1\}$ and $\alpha_2 \in \{0, 1, \ldots, c_2\}$, and by $F_0(\alpha_1, \alpha_2) = -(\min\{\alpha_1, 0\} + \min\{\alpha_2, 0\})$ if $\alpha_1 \in \{-P, \ldots, -1\}$ or $\alpha_2 \in \{-P, \ldots, -1\}$, where $P = \sum_{j=1}^{n} p_j$. The recursive relation is

$$F_k(\alpha_1, \alpha_2) = \max\{F_{k-1}(\alpha_1 - \ell, \alpha_2 + \ell - p_k) : \ell \in \Delta_k\},$$

where $\Delta_k = \{0, p_k\} \cup \left(\mathbb{N} \cap \left[\text{LB}_k, p_k - \text{LB}_k\right]\right)$, as in Section 3.4. The optimal objective value to an instance of RLP-SCHEDULING without upper bounds and with two periods is then $F_n(c_1, c_2)$. The time complexity of this algorithm is $O\left(P^2\, n\right)$.

We also establish the following result for this problem.

Lemma 12. *Problem* RLP-SCHEDULING *without upper bounds and with two periods admits an approximation algorithm only if* P = NP.

Proof: Consider the reduction from PARTITION constructed in the proof of Lemma 5. We have a YES instance of PARTITION if and only if the constructed instance of RLP-SCHEDULING has an optimal objective value of 0. Therefore, an approximation algorithm for RLP-SCHEDULING would correctly decide each instance of PARTITION. □

With free number of periods, it follows from Lemma 6 that RLP-SCHEDULING without upper bounds is NP-hard in the strong sense. Furthermore, unless P = NP, the problem does not admit an approximation algorithm. We close this section by observing that the DP algorithm and the complexity results established above remain valid even if each job must be executed without preemption.

4.5 There is only one job

We present Algorithm 4, which is adapted from Algorithm 2, for solving the problem RLP-SCHEDULING with a single job. In the description we omit the subscript referring to the job, and without loss of generality we assume that the release date is $r = 0$ and the deadline $\bar{d} = H$.

Algorithm 4 RLP-SCHEDULING with a single job

1: sort periods in non-increasing order of the available number of workers
2: **if** $p > H \cdot \text{UB}$ **then** return *the instance is infeasible*
3: **if** $p > \sum_{t=1}^{H} \min\{c_t, \text{UB}\}$ **then** return $p - \sum_{t=1}^{H} \min\{c_t, \text{UB}\}$
4: **if** $\text{LB} \cdot H \leq p$ **then** return 0
5: **else**
6: find the largest integer H' such that $\text{LB} \cdot H' \leq p$
7: **if** $p \leq \sum_{t=1}^{H'} \min\{c_t, \text{UB}\}$ **then** return 0
8: **else if** $p > H' \cdot \text{UB}$ **then** return *the instance is infeasible*
9: **else** return $p - \sum_{t=1}^{H'} \min\{c_t, \text{UB}\}$

Lemma 13. *Algorithm 4 solves the problem* RLP-SCHEDULING *with a single job in time* $O(H \ln H)$.

In Appendix C, we present a modified version of Algorithm 4 that runs in time $O(H)$ to solve the problem RLP-SCHEDULING with a single job that must be executed without preemption.

4.6 There are two jobs

A consequence of Lemma 8 is that the problem RLP-SCHEDULING with two jobs is NP-hard, even if the two jobs are identical. We now present a DP algorithm for solving instances of RLP-SCHEDULING with two jobs. Without loss of generality, we assume that the two jobs have the same time window. We define the value function $F_t(\beta_1, \beta_2)$ as the number of additional worker-periods needed when β_i workers in total are assigned to job i $(i = 1, 2)$ during periods $1, \ldots, t$. The initial condition is

$$F_1(\beta_1, \beta_2) = \max\{0,\ \beta_1 + \beta_2 - c_1\} \text{ for } \beta_j \in \{0, \text{LB}_j, \text{LB}_j + 1, \ldots, \text{UB}_j\}\ (j = 1, 2),$$

and $F_1(\beta_1, \beta_2) = +\infty$ otherwise. The recursive relation is

$$F_t(\beta_1, \beta_2) = \min\Big\{ \max\{0,\ \ell_1 + \ell_2 - c_t\} + F_{t-1}(\beta_1 - \ell_1, \beta_2 - \ell_2) : \\ \ell_j \in \{0, \text{LB}_j, \text{LB}_j + 1, \ldots, \text{UB}_j\} \text{ and } \ell_j \le \beta_j,\ j = 1, 2\Big\}.$$

The optimal objective value to an instance of RLP-SCHEDULING with two jobs is $F_H(p_1, p_2)$; if $F_H(p_1, p_2) = +\infty$ then the instance is infeasible. The time complexity of this algorithm is $O\left(P^2 H\right)$, where $P = \max\{p_1, p_2\}$.

We also establish the following non-approximation result for this problem.

Lemma 14. *Unless* P = NP*, the problem* RLP-SCHEDULING *with two jobs does not admit an approximation algorithm.*

Proof: The proof of this result is similar to that of Lemma 12, but then with reference to Lemma 8 instead of Lemma 5. □

When the two jobs must be executed without preemption, we mention that Algorithm 7 can be modified to solve the problem in time $O(H^4)$.

5. Problem RLP-SELECTION

Similarly as for Section 4, we observe that by solving RLP-SELECTION we can infer the solution to RLP-DECISION; as a consequence, all the negative results established in Section 3 remain valid for the problem RLP-SELECTION under the same conditions.

5.1 No time windows and no bound constraints

Unlike the problems RLP-DECISION and RLP-SCHEDULING, which are solved in linear time under this condition, the next result shows that the problem RLP-SELECTION is equivalent to the classic 0/1 knapsack problem.

Lemma 15. *The problem* RLP-SELECTION *is equivalent to the 0/1 knapsack problem when there are no time windows and no bound constraints, even if there is only one period.*

Proof: Given an instance of RLP-SELECTION with the above restrictions, we build an instance of the knapsack problem as follows: the capacity of the knapsack is $B = \sum_t^H c_t$, and there are n items. Item j has profit Q_j and weight p_j. Clearly, an optimal solution to this instance of the knapsack problem corresponds with an optimal solution to the instance of RLP-SELECTION and vice versa. Moreover, an instance of the knapsack problem can be seen as an instance of RLP-SELECTION with a single period. This completes the proof. □

The problem RLP-SELECTION without time windows and without bound constraints is therefore NP-hard in the ordinary sense. Furthermore, all known exact and approximation algorithms for knapsack problems are also valid for this special case of RLP-SELECTION. All these results remain valid even if each job must be executed without preemption.

5.2 No bound constraints

This case being a generalization of the one in Section 5.1, we infer that the problem RLP-SELECTION without bound constraints is at least NP-hard in the ordinary sense, even if there is only one period. If $H = 1$ then the problem RLP-SELECTION without bound constraints is equivalent to the knapsack problem, but when $H > 1$ this no longer holds, nor is the problem then equivalent to the classic multiple knapsack problem since this variant of RLP-SELECTION has time windows and a job can be scheduled in more than one time period. We propose a standard DP algorithm for this case. We assume that jobs are already sorted in non-increasing order of their deadlines; this can be done in time $O(n \ln n)$.

For ease of exposition, we add a dummy job 0 with $\bar{d}_0 \geq \bar{d}_1$ and $r_0 = p_0 = Q_0 = 0$. In this way, there is always an optimal solution that contains job 0. We use the following value function:

$$F_k(\alpha_1, \ldots, \alpha_H) \quad \equiv \quad \text{the maximum revenue obtained by considering jobs } \{0, 1, \ldots, k\} \text{ and using } \alpha_t \geq 0 \text{ workers in period } t.$$

The initial condition is $F_0(\alpha_1, \ldots, \alpha_H) = 0$ for $\alpha_t = 0, 1, \ldots, c_t$, and the recursive relation is formulated as follows:

$$F_k(\alpha_1, \ldots, \alpha_H) = \begin{cases} F_{k-1}(\alpha_1, \ldots, \alpha_H) & \text{if } \sum_{t=r_k+1}^{\bar{d}_k} \alpha_t < p_k \\ \max\{F_{k-1}(\alpha_1, \ldots, \alpha_H),\ Q_k + F_{k-1}(\bar{\alpha}_1, \ldots, \bar{\alpha}_H)\} & \text{otherwise,} \end{cases}$$

where the values $\bar{\alpha}_t$ are the output of Algorithm 5. The optimal objective value is $F_n(c_1, \ldots, c_H)$. Note that the jobs are indexed in non-increasing deadline, and that the DP algorithm actually schedules the jobs in reverse order. Algorithm 5 allows to avoid evaluating all possible assignments of the p_k joblets within the time window of job k, and is based on our result obtained in Section 3.2 that Algorithm 1 produces a feasible schedule for the decision version of this problem: the workload of input job k is scheduled as early as possible within its time window conditional on $(c_t - \alpha_t)$ resource units already being allocated to jobs $k+1, \ldots, n$ $(t = 1, \ldots, H)$ on computing $F_k(\alpha_1, \ldots, \alpha_H)$.

Algorithm 5 Compute $(\bar{\alpha}_1, \bar{\alpha}_2, \ldots, \bar{\alpha}_H)$

Input: vector $(\alpha_1, \alpha_2, \ldots, \alpha_H)$, job k and revenue q
Output: vector $(\bar{\alpha}_1, \bar{\alpha}_2, \ldots, \bar{\alpha}_H)$

```
t = 0 and Δ = p_k
while t ≤ H do
    if t ∉ {r_k + 1, ..., d̄_k} then ᾱ_t = α_t
    else
        ᾱ_t = max{α_t − Δ, 0}
        Δ = max{Δ − α_t, 0}
    end if
    t ← t + 1
```

The time complexity of this DP procedure is $O(C\,n + n \ln n)$, where $C = \prod_{t=1}^{H} c_t$; this is pseudo-polynomial.

When jobs must be executed without preemption, we establish the following non-approximability result.

Lemma 16. *Unless* P = NP, *the problem* RLP-SELECTION *without bound constraints and without preemption does not admit an* FPTAS.

Proof: We still consider the reduction from SEQUENCING WITH RELEASE TIMES AND DEADLINES constructed in the proof of Lemma 3, and in addition each job $j \in N$ has revenue $Q_j = 1$. If we have a YES instance of SEQUENCING WITH RELEASE TIMES AND DEADLINES then the constructed instance of RLP-SELECTION has an optimal objective value of n; otherwise, it is at most $n-1$. Therefore, by taking $\epsilon = \frac{1}{n+1}$ any FPTAS can be used to distinguish between these two situations in polynomial time. □

5.3 No lower bounds

The problem RLP-SELECTION without lower bounds is also a generalization of the problem studied in Section 5.1, and so it is at least NP-hard in the ordinary sense, even if there is only one period. When there is exactly one period, the problem is equivalent to the knapsack problem (by eliminating the jobs that do not satisfy the upper bounds). When the number of periods $H > 1$, the DP algorithm proposed in Section 5.2 can be modified to handle this problem, with the necessary changes described in Section 3.2 to Algorithm 1 applied to Algorithm 5. Again we observe that without preemption, the inclusion of non-trivial upper bounds does not make the problem essentially more difficult, contrary to lower bounds (see Section 5.4).

We close this section by noting that with a non-preemption constraint, RLP-SELECTION becomes NP-hard in the strong sense, and hence does not admit an FPTAS unless P = NP.

5.4 No upper bounds

Lemma 5 implies that RLP-SELECTION without upper bounds is NP-hard, even if there are only two periods and all jobs are splittable. Note that with only one period, this problem is equivalent to the knapsack problem and hence can be solved by any known knapsack algorithm.

We now consider the case when there is more than one period. The DP algorithm proposed in Section 5.2 can be modified to handle this problem by examining all possible $\bar{\alpha}_t$ at each recursion. For simplicity, we present the DP algorithm for two periods. We also add a dummy job 0 with $\bar{d}_0 < \bar{d}_1$, $r_0 = \bar{d}_0$, $p_0 = 0$ and $Q_0 = 0$. We use the DP value function $F_k(\alpha_1, \alpha_2)$, which represents the maximum revenue that can be collected by considering jobs $\{0, 1, \ldots, k\}$ and α_t workers in period t ($t = 1, 2$). The initial condition of the DP recursion is $F_0(\alpha_1, \alpha_2) = 0$ for $\alpha_1 = 0, 1, \ldots, c_1$ and $\alpha_2 = 0, 1, \ldots, c_2$, and $F_0(\alpha_1, \alpha_2) = -\infty$ if $\alpha_1 < 0$ or $\alpha_2 < 0$. The recursive relation is as follows: if $\alpha_1 + \alpha_2 < p_k$ then $F_k(\alpha_1, \alpha_2) = F_{k-1}(\alpha_1, \alpha_2)$; otherwise,

$$F_k(\alpha_1, \alpha_2) = \max \left\{ \begin{array}{l} F_{k-1}(\alpha_1, \alpha_2), \\ Q_k + F_{k-1}(\alpha_1 - p_k, \alpha_2) \text{ if } [r_k, \bar{d}_k] = [0, 1], \\ Q_k + F_{k-1}(\alpha_1, \alpha_2 - p_k) \text{ if } [r_k, \bar{d}_k] = [1, 2], \\ Q_k + F_{k-1}(\alpha_1 - \ell, \alpha_2 + \ell - p_k) : \ell \in \Delta_k \\ \qquad\qquad\qquad\qquad \text{if } r_k = 0 \text{ and } \bar{d}_k = 2 \end{array} \right\},$$

where $\Delta_k = \{0, p_k\} \cup \left(\mathbb{N} \cap [\text{LB}_k, p_k - \text{LB}_k]\right)$. The optimal objective value is given by $F_n(c_1, c_2)$. The time complexity of this DP algorithm is $O(C^2 n)$, where $C = \max\{c_1, c_2\}$. By the same reasoning as in Section 3.4, this DP recursion can be generalized for any constant H: the total number of

partitions $(\ell_1, \ell_2, \ldots, \ell_H)$ of p_k, with $\sum_{i=1}^{H} \ell_i = p_k$, is then constant in the number of jobs, and therefore the resulting algorithm would still run in pseudo-polynomial time.

It is well known that the knapsack problem admits an FPTAS, therefore we can conclude that RLP-SELECTION without upper bounds admits an FPTAS when $H = 1$. Conversely, we can establish the following non-approximation result when $H > 1$.

Lemma 17. *Unless* P = NP*, the problem* RLP-SELECTION *with two periods and no upper bounds does not admit an* FPTAS*.*

Proof: We consider the reduction from PARTITION set up in the proof of Lemma 5 where in addition each job j has revenue $Q_j = 1$. If we have a YES instance of PARTITION then the optimal objective value is $2m$; otherwise, it is at most $2m - 1$. Therefore, an FPTAS for RLP-SELECTION with two periods and no upper bounds can be used to distinguish between these two situations in polynomial time. □

When the non-preemption constraint is included into the problem statement, both the positive as well as the negative results of this section can be seen to carry over.

5.5 There is only one job

There are two options in this case, namely the selection (and execution) of the job, or no selection. The problem can be solved by an adapted version of Algorithm 2, where in lines 3 and 6 we return Q (the revenue of the unique job), and in lines 2 and 7 we return 0 (meaning that the job is not selected). By applying similar modifications to Algorithm 6, we can solve RLP-SELECTION with only one job and without preemption.

5.6 There are two jobs

Lemma 8 implies that RLP-SELECTION with two jobs is NP-hard, even if the two jobs are identical. We now describe an algorithm for solving this problem. We refer to the two jobs as job 1 and job 2. Four options occur: (1) none of the two jobs is selected; (2) job 1 is selected and not job 2; (3) job 2 is selected and not job 1; and (4) both job 1 and job 2 are executed. The first option leads to an objective value of zero. The second option gives rise to an objective value of either zero or Q_1, and the third to either zero or Q_2, which can be found via the procedure described in Section 5.5. The last option, which is investigated only if option (2) leads to Q_1 and option (3) to Q_2, generates a revenue of either zero or $Q_1 + Q_2$ and can be scanned with the DP algorithm described for RLP-DECISION with two jobs. Since we have four possibilities, and since each case requires an algorithm that is at most pseudo-polynomial, we conclude that RLP-SELECTION with two jobs can be solved in pseudo-polynomial time. The described procedure remains valid even if each job must be executed without preemption.

6. Conclusions

In this paper, we have defined a generic resource scheduling problem that was inspired by tactical capacity planning in project-based environments. We contribute to a better understanding of this problem by a structured complexity analysis of the underlying combinatorial optimization problem. In the process, we have recognized a number of links with existing work in equipment maintenance and MTO and ETO planning, but also in operational and multiprocessor scheduling. Our results are useful to further the understanding of the structural properties of resource loading, and the

proposed algorithms can be implemented as subroutine within any framework for solving practical resource loading problems.

Further work in this area should indeed focus on the incorporation of the practical aspects of particular planning cases. In one particular aircraft line maintenance problem, for instance, we learned that the intensities are required to be monotone, meaning that a job is executed either with non-decreasing or with non-increasing intensities. This would mean, for instance, that one cannot start a job with one person, then continue with two persons and finish again with one person; this monotonicity constraint implies that the jobs are executed without preemption. It is remarkable that similar monotonicity constraints have been identified as dominant properties in some related papers (see Hendel et al., 2014; Sadykov, 2012), and this aspect certainly deserves further research. Resource loading as described in this paper can also be seen as a variant of the multiple knapsack problem, in which items can be spread across different knapsacks. The bin packing problem is closely related to the multiple knapsack problem, and the possibility of splitting jobs across multiple bins has recently also been studied (see Casazza and Ceselli, 2014); synergies might also be explored in this area.

Appendix

A. RLP-DECISION with a single non-preemptible job

We present Algorithm 6 for solving RLP-DECISION with a single job that must be executed without preemption. Without loss of generality, we assume $c_t \geq \text{LB}$, $\forall t \in H$.

Algorithm 6 RLP-DECISION with a single non-preemptible job

1: **if** $p > \sum_{t=1}^{H} \min\{c_t, \text{UB}\}$ **then** return NO
2: **if** $\text{LB} \cdot H \leq p$ **then** return YES
3: **else**
4: **for** $t = 1$ to H **do**
5: find the largest integer H_t with $t \leq H_t \leq H$ such that $\text{LB} \cdot (H_t - t + 1) \leq p$
6: **if** $p \leq \sum_{\ell=t}^{H_t} \min\{c_\ell, \text{UB}\}$ **then** return YES
7: return NO

By observing that H_t in line 5 can be obtained as $\min\{H, \lfloor \frac{p}{\text{LB}} \rfloor + t - 1\}$ (assuming $p \geq \text{LB}$), and with a careful monitoring of the sums in line 6, the following result can be seen to hold.

Lemma 18. *Algorithm 6 solves the problem* RLP-DECISION *with a single job that must be executed without preemption in time* $O(H)$.

B. RLP-DECISION with two non-preemptible jobs

We use t_1^s and t_1^e (respectively t_2^s and t_2^e) to refer to the first and the last period in which job 1 (respectively job 2) is executed. Note that for a feasible schedule, we have $r_1 \leq t_1^s \leq t_1^e \leq \bar{d}_1$ and $r_2 \leq t_2^s \leq t_2^e \leq \bar{d}_2$. Without loss of generality, we assume that $r_1 \leq r_2$. We present Algorithm 7 for solving RLP-DECISION with two jobs that must be executed without preemption, and we formulate the following result.

Lemma 19. *Algorithm 7 solves the problem* RLP-DECISION *with two non-preemptible jobs in time* $O(H^4)$.

Algorithm 7 RLP-DECISION with two non-preemptible jobs

run Algorithm 6 for job 1 and if it outputs NO then return NO
run Algorithm 6 for job 2 and if it outputs NO then return NO
for $t_1^s = r_1 + 1$ to $\bar{d}_1$ **do**
  for $t_1^e = t_1^s$ to $\bar{d}_1$ **do**
    if $\text{LB}_1 \cdot (t_1^e - t_1^s + 1) \leq p_1 \leq \sum_{t_1=t_1^s}^{t_1^e} \min\{c_{t_1}, \text{UB}_1\}$ **then**
      for $t_2^s = r_2 + 1$ to $\bar{d}_2$ **do**
        for $t_2^e = t_2^s$ to $\bar{d}_2$ **do**
          if $\text{LB}_2 \cdot (t_2^e - t_2^s + 1) \leq p_2 \leq \sum_{t_2=t_2^s}^{t_2^e} \min\{c_{t_2}, \text{UB}_2\}$ **then**
            if $t_1^e < t_2^s$ or $t_2^e < t_1^s$ **then** return YES
            else if $t_1^s < t_2^s$ and $c_t \geq \text{LB}_1 + \text{LB}_2$ for $t_2^s \leq t \leq \min\{t_1^e, t_2^e\}$ **then**
              $A_1 = \sum_{t=t_1^s}^{t_2^s-1} \min\{c_t, \text{UB}_1\}$
              $A_2 = \sum_{t=t_2^s}^{\min\{t_1^e,t_2^e\}} \min\{c_t, \text{UB}_1 + \text{UB}_2\}$
              $A_3 = \begin{cases} \sum_{t=t_1^e+1}^{t_2^e} \min\{c_t, \text{UB}_2\} & \text{if } t_2^e \geq t_1^e \\ \sum_{t=t_2^e+1}^{t_1^e} \min\{c_t, \text{UB}_1\} & \text{if } t_1^e > t_2^e \end{cases}$
              if $p_1 + p_2 \leq A_1 + A_2 + A_3$ **then** return YES
            else if $t_2^s \leq t_1^s$ and $c_t \geq \text{LB}_1 + \text{LB}_2$ for $t_1^s \leq t \leq \min\{t_1^e, t_2^e\}$ **then**
              $A_1 = \sum_{t=t_2^s}^{t_1^s-1} \min\{c_t, \text{UB}_2\}$
              $A_2 = \sum_{t=t_1^s}^{\min\{t_1^e,t_2^e\}} \min\{c_t, \text{UB}_1 + \text{UB}_2\}$
              $A_3 = \begin{cases} \sum_{t=t_1^e+1}^{t_2^e} \min\{c_t, \text{UB}_2\} & \text{if } t_2^e \geq t_1^e \\ \sum_{t=t_2^e+1}^{t_1^e} \min\{c_t, \text{UB}_1\} & \text{if } t_1^e > t_2^e \end{cases}$
              if $p_1 + p_2 \leq A_1 + A_2 + A_3$ **then** return YES
return NO

Proof: The correctness of Algorithm 7 is guaranteed by the complete enumeration of all possible cases and its time complexity results from the nested loops. □

C. RLP-SCHEDULING with a single non-preemptible job

Algorithm 8 runs in time $O(H)$ and solves the problem RLP-SCHEDULING with a single job that must be executed without preemption.

Acknowledgements

The project leading to these results has been partially supported by the bilateral scientific cooperation project BIL10/10 between KU Leuven (Belgium) and Tsinghua University (China). Fabrice Talla Nobibon gratefully acknowledges the Fund for Scientific Research - Flanders (FWO-Vlaanderen) for his postdoctoral fellowship. Zhenbo Wang also received support from project NSFC No. 11371216.

Algorithm 8 RLP-SCHEDULING with a single non-preemptible job

if $p > H \cdot \text{UB}$ **then** return *the instance is infeasible*
else if $p > \sum_{t=1}^{H} \min\{c_t, \text{UB}\}$ **then** return $p - \sum_{t=1}^{H} \min\{c_t, \text{UB}\}$
else if $\text{LB} \cdot H \leq p$ **then** return 0
else
    $z = +\infty$
    for $t = 1$ to H **do**
        find the largest integer H_t with $t \leq H_t \leq H$ such that $\text{LB} \cdot (H_t - t + 1) \leq p$
        if $p \leq \sum_{\ell=t}^{H_t} \min\{c_\ell, \text{UB}\}$ **then** $z = 0$
        else if $p \leq (H_t - t + 1) \cdot \text{UB}$ **then** $z = \min\left\{z, p - \sum_{\ell=t}^{H_t} \min\{c_\ell, \text{UB}\}\right\}$
    if $z = +\infty$ **then** return *the instance is infeasible*
    else return z

References

Adler, P., Mandelbaum, A., Nguyen, V., Schwerer, E., 1995. From project to process management: An empirically-based framework for analyzing product development time. Management Science 41 (3), 458–484.

Ahuja, R., Magnanti, T., Orlin, J., 1993. Network Flows: Theory, Algorithms, and Applications. Prentice Hall.

Alfieri, A., Tolio, T., Urgo, M., 2011. A project scheduling approach to production planning with feeding precedence relations. International Journal of Production Research 49 (4), 995–1020.

Baptiste, P., Le Pape, C., Nuijten, W., 1999. Satisfiability tests and time-bound adjustments for cumulative scheduling problems. Annals of Operations Research 92, 305–333.

Bar-Noy, A., Guha, S., Naor, J., Schieber, B., 2001. Approximating the throughput of multiple machines in real-time scheduling. SIAM Journal on Computing 31 (2), 331–352.

Beliën, J., Demeulemeester, E., Cardoen, B., 2012. Improving workforce scheduling of aircraft line maintenance at sabena technics. Interfaces 42 (4), 352–364.

Casazza, M., Ceselli, A., 2014. Mathematical programming algorithms for bin packing problems with item fragmentation. Computers & Operations Research, forthcoming.

Cormen, T., Leiserson, C., Rivest, R., Stein, C., 2005. Introduction to Algorithms. The MIT Press.

De Boer, R., 1998. Resource-constrained multi-project management – a hierarchical decision support system. Ph.D. thesis, University of Twente, The Netherlands.

Drozdowski, M., 2009. Scheduling for Parallel Processing. Springer.

Ebben, M., Hans, E., Olde Weghuis, F., 2005. Workload based order acceptance in job shop environments. OR Spectrum 27 (1), 107–122.

Engels, D., Karger, D., Kolliopoulos, S., Sengupta, S., Uma, R., Wein, J., 2003. Techniques for scheduling with rejection. Journal of Algorithms 49, 175–191.

Gademann, N., Schutten, M., 2005. Linear-programming-based heuristics for project capacity planning. IIE Transactions 37, 153–165.

Garey, M., Johnson, D., 1979. Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman and Co.

Guerrero, H., Kern, G., 1998. How to more effectively accept and refuse orders. Production and Inventory Management Journal 4, 59–62.

Günther, E., König, F., Megow, N., 2014. Scheduling and packing malleable and parallel tasks with precedence constraints of bounded width. Journal of Combinatorial Optimization, forthcoming.

Hans, E., 2001. Resource loading by branch-and-price techniques. Ph.D. thesis, University of Twente, The Netherlands.

Haugland, D., Eleyat, M., Hetland, M., 2011. The maximum flow problem with minimum lot sizes. In: Proceedings of the 2nd International Conference on Computational Logistics (ICCL). Vol. 6971 of LNCS. pp. 170–182.

Hayes, R., Wheelright, S., 1979. Link manufacturing process and product life cycles. Harvard Business Review (January-February), 133–140.

Hendel, Y., Kubiak, W., Trystram, D., 2014. Scheduling semi-malleable jobs to minimize mean flow time. Journal of Scheduling, forthcoming.

Herbots, J., Herroelen, W., Leus, R., 2007. Dynamic order acceptance and capacity planning on a single bottleneck resource. Naval Research Logistics 54 (8), 874–889.

Jansen, K., 2004. Scheduling malleable parallel tasks: an asymptotic fully polynomial time approximation scheme. Algorithmica 39 (1), 59–81.

Jansen, K., Porkolab, L., 2002. Linear-time approximation schemes for scheduling malleable parallel tasks. Algorithmica 32 (3), 507–520.

Kerzner, H., 1998. Project Management. A Systems Approach to Planning, Scheduling and Controlling. Wiley.

Keskinocak, P., Tayur, S., 2004. Due date management policies. In: Simchi-Levi, D., Wu, S., Shen, Z. (Eds.), Handbook of Quantitative Supply Chain Analysis: Modeling in the E-Business Era. Kluwer, Ch. 12.

Kim, Y.-D., Shim, S.-O., Kim, S.-B., Choi, Y.-C., Yoon, H., 2004. Parallel machine scheduling considering a job-splitting property. International Journal of Production Research 42, 4531–4546.

Kis, T., 2005. A branch-and-cut algorithm for scheduling of projects with variable-intensity activities. Mathematical Programming 103, 515–539.

Kogan, K., Shtub, A., 1999. Scheduling projects with variable-intensity activities: The case of dynamic earliness and tardiness costs. European Journal of Operational Research 118, 65–80.

Larson, E., Gobeli, D., 1989. Significance of project management structure on development success. IEEE Transactions on Engineering Management 36 (2), 119–125.

Leachman, R., 1983. Multiple resource leveling in construction systems through variation of activity intensities. Naval Research Logistics Quarterly 30 (2), 187–198.

Leachman, R., Dincerler, A., Kim, S., 1990. Resource-constrained scheduling of projects with variable-intensity activities. IIE Transactions 22 (1), 31–39.

Masmoudi, M., 2011. Tactical and operational project planning under uncertainties: application to helicopter maintenance. Ph.D. thesis, University of Toulouse, France.

Mestry, S., Damodaran, P., Chen, C., 2011. A branch and price solution approach for order acceptance and capacity planning in make-to-order operations. European Journal of Operational Research 211 (3), 480–495.

Moore, J., 1968. An n job, one machine sequencing algorithm for minimizing the number of late jobs. Management Science 15 (1), 102–109.

Mounié, G., Rapine, C., Trystram, D., 2007. A 3/2-dual approximation algorithm for scheduling independent monotonic malleable tasks. SIAM Journal on Computing 37 (2), 401–412.

Pinedo, M., 2008. Scheduling: Theory, Algorithms, and Systems, 3rd Edition. Prentice Hall.

Pochet, Y., Wolsey, L., 2006. Production Planning by Mixed Integer Programming. Springer.

Roundy, R., Chen, D., Chen, P., Cakanyildirim, M., Freimer, M., Melkonian, V., 2005. Capacity-driven acceptance of customer orders for a multi-stage batch manufacturing system: Models and algorithms. IIE Transactions 37, 1093–1105.

Sadykov, R., 2012. A dominant class of schedules for malleable jobs in the problem to minimize the total weighted completion time. Computers & Operations Research 39, 1265–1270.

Serafini, P., 1996. Scheduling jobs on several machines with the job splitting property. Operations Research 44, 617–628.

Slotnick, S., 2011. Order acceptance and scheduling: A taxonomy and review. European Journal of Operational Research 1, 1–11.

Slotnick, S., Morton, T., 2007. Order acceptance with weighted tardiness. Computers & Operations Research 34 (10), 3029–3042.

Smeulders, B., 2011. Exploring the grey zone between planning and scheduling for the metals industry. Master's thesis, KU Leuven, Belgium.

Speranza, M., Vercellis, C., 1993. Hierarchical models for multi-project planning and scheduling. European Journal of Operational Research 64, 312–325.

Suerie, C., Stadtler, H., 2003. The capacitated lot-sizing problem with linked lot sizes. Management Science 49 (8), 1039–1054.

Thielen, C., Westphal, S., 2012. Complexity and approximability of the maximum flow problem with minimum quantities. Report in Wirtschaftsmathematik (WIMA Report) 143, Technische Universität Kaiserslautern, Germany.

Weglarz, J., 1981. Project scheduling with continuously-divisible, doubly constrained resources. Management Science 27, 1040–1053.

Wheelright, S., Clark, K., 1992. Creating project plans to focus product development. Harvard Business Review (March-April), 70–82.

Wullink, G., Gademann, A., Hans, E., van Harten, A., 2004. Scenario-based approach for flexible resource loading under uncertainty. International Journal of Production Research 42 (24), 5079–5098.

Xing, W., Zhang, J., 2000. Parallel machine scheduling with splitting jobs. Discrete Applied Mathematics 103, 259–269.

FACULTY OF ECONOMICS AND BUSINESS
Naamsestraat 69 bus 3500
3000 LEUVEN, BELGIË
tel. + 32 16 32 66 12
fax + 32 16 32 67 91
info@econ.kuleuven.be
www.econ.kuleuven.be